A SYSTEM AND METHOD USING A CIRCULAR BUFFER FOR DETECTING PACKET LOSS IN STREAMING APPLICATIONS

This invention relates to the detection of packet loss in Internet streaming. More particularly, this invention relates to a system and method for using a circular buffer, implemented by a chain of buffers forming a circle, for packet loss detection in Internet streaming. Most particularly, this invention relates to a system and method for dynamically adapting packet loss detection latency, which is determined by the size of the chain, to network conditions and application requirements that achieves reasonable detection accuracy and is easy to implement.

In Internet streaming applications, when an important packet gets lost, such as a packet belonging to an I-frame in video or the base-layer in scalable coding, the receiver may ask the sender to retransmit this lost packet. In order to send a retransmission request promptly, the receiver must have means to timely detect packet losses. Currently, packet loss detection is done using either a timer or timing windows.

In the transmission control (TCP) protocol, a timer is used for loss detection. When a packet is sent, a timer with a timeout value is set by the sender for that packet. If the timer expires before the acknowledgement of the packet is received, the packet is declared as lost and resent by the sender.

In most streaming applications, loss detection is done by using timing windows. A timing window (or more precisely a table) has a fixed number of binary entries. Each entry indicates a packet 's status (0: lost, 1: received). At a certain point of time, the first entry in this window is associated with a packet that is identified by a sequence number (such as the sequence number of a real-time transport protocol (RTP) packet). The subsequent window entries are associated with packets having higher sequence numbers in sequential order. Therefore, the space of packet sequence numbers can be viewed as being divided into blocks, each block being associated with a window at a certain point of time.

Presently, two such timing windows associated with two consecutive blocks of packets are used for packet loss detection. In the beginning, all entries are marked as "0". When a packet is received, the corresponding window entry is marked as "1". When a packet is received that has a sequence number that goes beyond the first window, the corresponding second window entry is marked. As soon as a packet is transmitted with a sequence number that goes beyond even the second window, the first window is closed, and its entries are checked. Packets associated with entries that remain marked as "0", are declared as lost. A consecutive new window is opened right after the second window and the detection process resumes.

The timer method can be applied only to TCP-like protocols that can measure packet round-

trip time in order to properly set the timeout value of the timer. In streaming applications, most of the time only unidirectional media streams are generated. The timer method is not applicable to these cases. Instead, timing window methods are used. However, there are limitations with the timing window methods:

- The window size is fixed - there is no built-in mechanism for window size adaptation, which is desirable when network conditions and application requirements (e.g. delay) change.
- Non-uniform loss detection latency for different losses occurs - the lost detection latency lies in a range between T and ~ 2T, where T is the average period of the timing window, such that when a loss is associated with the first entry of the window, the detection latency is 2T, while it is T if the loss is associated with the last entry of the window.

Thus, there is a need for a loss detection method that allows adaptive loss detection latency, as well as a uniform loss detection latency. The system and method of the present invention comprises:

- a circular chain of buffers having a chain size that is adjustable according to network conditions and application requirements, thereby providing adaptive loss detection latency; and
- a circular chain of buffers having a fixed chain size, thereby providing uniform detection latency when the chain size is fixed.

Shortening the latency can increase the chance of recovering a lost packet. Having a uniform latency may imply an equal recovery chance for all losses. Therefore having an equal latency for all loss detection may be desirable for many streaming applications.

The implementation overhead of the circular buffer of the present invention is low and can be less than the timing window method.

FIG. 1a illustrates a preferred embodiment of the circular buffer structure of the present invention.

FIG. 1b illustrates the structure of each of the buffers in the circular buffer illustrated in FIG. 1a.

FIG. 2a illustrates an algorithm of a preferred that implements the circular buffer structure illustrated in FIGs. 1a-b.

FIG. 2b illustrates a flow chart of the algorithm illustrated in FIG. 2a.

FIG. 3 illustrates a preferred embodiment of an algorithm for adapting the structure of the circular buffer chain based on network characteristics.

Referring now to FIB. 1a, buffers 1 through m form a circularly linked list of m buffers B_i 10 where $i= 1, ..., m$ each of whose structure 12 is shown in FIG. 1b. As shown in FIG. 1a, m is the length of the circular chain that determines the loss detection latency, and it can be adapted to network conditions and application requirements. P 11 is a pointer that circulates through the chain, pointing to each buffer in turn. Each buffer B_i 10 in the chain comprises two fields, F_1 13 and F_2 14. F_1 13 stores a pointer to the next buffer. F_2 14 stores a sequence number s of a packet that may get lost.

In streaming applications, packets are supposed sent in the order of packet sequence number. In a no-loss and ideal world, an arriving packet always has a sequence number one higher than the previous. If a packet arrives out of order, or is lost, then a hole or gap is observed in the sequence numbers of the received packets. Whenever a hole (could be a hole that spans more than one consecutive number) is observed, potentially, this hole may indicate one or more lost packets. However, out-of-order packet delivery is common to Internet because each packet can take a different path through the network and an earlier numbered packet may take longer to arrive than

a later numbered packet. An application cannot make a loss declaration immediately after observing a hole in the sequence of arriving packets. The application has to wait and see whether this hole is just an incident of out-of-order delivery. The circular buffer method provides a way to determine if a loss declaration can be made.

FIG. 2a illustrates C programming language code for a preferred embodiment of an algorithm for accomplishing the method of the present invention. FIG. 2b is a flow chart of the algorithm illustrated in FIG. 2a. The pointer P 12 circulates through the chain of buffers B_i 11, the circulation being driven by receipt of a packet at step 20. The sequence number of the received packet is checked against that of the current maximum sequence number already received s at step 21, and if it is less than the current maximum sequence number received it is an out of order packet.

If it is not an out of order packet, at step 22 a hole in the sequence is checked for and when a hole in the sequence is observed the following steps are performed:

a. If the buffer at which P is pointing contains a non-received packet (P->$F2$ is not zero at step 24) then at step 25 the packet with the sequence number P->$F2$ is declared lost.

b. Then, regardless of whether or not P was pointing at a non-received packet, at step 26 the current maximum sequence number is incremented by one and stored in the current buffer and P is updated to point to the next buffer in sequence, i.e., P=P->$F1$.

c. The number of buffers that fall in the hole is decremented by 1 at step 27 and steps a-c are repeated until the remaining number of buffers is zero.

Thus, all the sequence numbers that fall in the hole are stored in the circular buffer chain, with each number occupying one buffer.

When a hole is not observed, the following steps are performed:

d. At step 28 if the buffer at which *P* is pointing contains a non-received packet (*P->F2* is not zero) then at step 29 the packet with the sequence number *P->F2* is declared lost and the sequence number stored in the buffer is set to zero

e. Whether or not *P->F2* points at a non-received packet, at step 30 *P* is updated to point to the next buffer in sequence.

When all the processing associated with an in order packet is completed:

f. At step 31 the current maximum sequence number is set to that of the received packet.

If a packet arrives out of order (having a sequence number that is earlier than the current received maximum sequence number *s*):

g. the received packet number is compared with the numbers stored in the circular buffer, and the corresponding record in the buffer is cleaned, i.e., set to zero at step 32.

Thus, the detection latency is determined by the size of the buffer chain *m*, because the loss declaration is only made when the pointer re-visits a non-empty buffer, i.e., when F_2 is non-zero.

As illustrated in FIG. 3, the chain size *m*, that determines the detection latency can be adapted. For example, in a preferred embodiment, initially $m = 4$. If the observed false declaration rate is higher than a given threshold, i.e.,

$$false\_rate > TOLERABLE\_RATE$$

the length may be too short and may need to be lengthened by inserting a new buffer and adjusting *m* correspondingly 36. The greater the length of the network path, i.e., number of links traversed, the larger the *m* that is needed, because when a packet

traverses a longer network path, there is a greater likelihood of out-of-order delivery occurring. The larger value of m decreases the likelihood of the pointer P encountering a delivered but out-of-order packet in a buffer as P circulates through the buffer chain B_i 11.

The *success_rate* is initially declared to be a pre-determined *EXPECT_RATE* and adjusted thereafter to be

$$success\_rate = \frac{declared\_losses - falsely\_declared\ losses}{declared\_losses}$$

$$false\_rate = 1 - success\_rate$$

and if the *success_rate* is too high, i.e., if

$$success\_rate > EXPECT\_RATE$$

the length of the buffer chain may be too long and may need to be shortened by deleting a buffer as illustrated in FIG. 3.

This present invention can be used in the implementation of multimedia players that play media from networked storage. Or it can be used by any type of multimedia receiver that wants to use retransmission as an error-recovery means, therefore need to perform packet loss detection. Finally, it can be used by transport control protocol implementations that the packet loss detection is done at the receiver side.

The methods and systems of the present invention, as described above and shown in the drawings, provide for a circular buffer that allows an adaptive latency detection time or a fixed latency detection time. It will be apparent to those skilled in the art that various modifications and variations can be made in the method and system of the present invention without departing from the spirit or scope of the invention.

Thus, it is intended that the present invention includes modifications and variations that are within the scope of the appended claims and their equivalents.